WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Tuesday, April 20, 2004

Hide?	Set Name	Query	Hit Count
		DB=USPT,PGPB; PLUR=YES; OP=ADJ	
☐	L12	L11 and index$	6
☐	L11	L6 and (scratch near register)	13
		DB=JPAB; PLUR=YES; OP=ADJ	
☐	L10	L6	0
		DB=EPAB,DWPI; PLUR=YES; OP=ADJ	
☐	L9	L6	1
		DB=TDBD; PLUR=YES; OP=ADJ	
☐	L8	L6	0
		DB=USPT,PGPB; PLUR=YES; OP=ADJ	
☐	L7	L6	94
		DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L6	L4 and ((index$ near2 register) or scratch)	95
☐	L5	L4 and (index$ near2 register)	70
☐	L4	L3 and (allocat$ near register)	384
☐	L3	instrument$ or profil$	1148327
		DB=USPT,PGPB; PLUR=YES; OP=ADJ	
☐	L2	((714/34 |714/35 |714/36 |714/37 |714/38 |714/39)!.CCLS.)	1912
☐	L1	((717/124 |717/127 |717/128 |717/129 |717/130 |717/131 |717/132 |717/133 |717/141 |717/142 |717/143 |717/144 |717/145 |717/155 |717/156 |717/158 |717/159)!.CCLS.)	2037

END OF SEARCH HISTORY

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 6 of 6 returned.

☐ 1. Document ID: US 20040062246 A1

Using default format because multiple data bases are involved.

L12: Entry 1 of 6 File: PGPB Apr 1, 2004

PGPUB-DOCUMENT-NUMBER: 20040062246
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20040062246 A1

TITLE: High performance network interface

PUBLICATION-DATE: April 1, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Boucher, Laurence B.	Saratoga	CA	US	
Blightman, Stephen E. J.	San Jose	CA	US	
Craft, Peter K.	San Francisco	CA	US	
Higgen, David A.	Saratoga	CA	US	
Philbrick, Clive M.	San Jose	CA	US	
Starr, Daryl D.	Milpitas	CA	US	

US-CL-CURRENT: 370/392; 709/236, 709/250

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Ir

☐ 2. Document ID: US 20040030745 A1

L12: Entry 2 of 6 File: PGPB Feb 12, 2004

PGPUB-DOCUMENT-NUMBER: 20040030745
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20040030745 A1

TITLE: Method and apparatus for distributing network traffic processing on a multiprocessor computer

PUBLICATION-DATE: February 12, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Boucher, Laurence B.	Saratoga	CA	US	

Blightman, Stephen E. J.	San Jose	CA	US
Craft, Peter K.	San Francisco	CA	US
Higgen, David A.	Saratoga	CA	US
Philbrick, Clive M.	San Jose	CA	US
Starr, Daryl D.	Milpitas	CA	US

US-CL-CURRENT: 709/203

ABSTRACT:

An intelligent network interface card (INIC) or communication processing device (CPD) works with a host computer for data communication. The device provides a fast-path that avoids protocol processing for most messages, greatly accelerating data transfer and offloading time-intensive processing tasks from the host CPU. The host retains a fallback processing capability for messages that do not fit fast-path criteria, with the device providing assistance such as validation even for slow-path messages, and messages being selected for either fast-path or slow-path processing. A context for a connection is defined that allows the device to move data, free of headers, directly to or from a destination or source in the host. The context can be passed back to the host for message processing by the host. The device contains specialized hardware circuits that are much faster at their specific tasks than a general purpose CPU. A preferred embodiment includes a trio of pipelined processors devoted to transmit, receive and utility processing, providing full duplex communication for four Fast Ethernet nodes.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Ir

---

☐ 3. Document ID: US 20040003126 A1

L12: Entry 3 of 6 File: PGPB Jan 1, 2004

PGPUB-DOCUMENT-NUMBER: 20040003126
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20040003126 A1

TITLE: TCP/IP offload network interface device

PUBLICATION-DATE: January 1, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Boucher, Laurence B.	Saratoga	CA	US	
Blightman, Stephen E. J.	San Jose	CA	US	
Craft, Peter K.	San Francisco	CA	US	
Higgen, David A.	Saratoga	CA	US	
Philbrick, Clive M.	San Jose	CA	US	
Starr, Daryl D.	Milpitas	CA	US	

US-CL-CURRENT: 709/250; 719/321

ABSTRACT:

An intelligent network interface card (INIC) or communication processing device (CPD) works with a host computer for data communication. The device provides a fast-path that avoids protocol processing for most messages, greatly accelerating data transfer and offloading time-intensive processing tasks from the host CPU. The host retains a fallback processing capability for messages that do not fit fast-path criteria, with the device providing assistance such as validation even for slow-path messages, and messages being selected for either fast-path or slow-path processing. A context for a connection is defined that allows the device to move data, free of headers, directly to or from a destination or source in the host. The context can be passed back to the host for message processing by the host. The device contains specialized hardware circuits that are much faster at their specific tasks than a general purpose CPU. A preferred embodiment includes a trio of pipelined processors devoted to transmit, receive and utility processing, providing full duplex communication for four Fast Ethernet nodes.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | In

---

☐ 4. Document ID: US 20020083425 A1

L12: Entry 4 of 6 File: PGPB Jun 27, 2002

PGPUB-DOCUMENT-NUMBER: 20020083425
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20020083425 A1

TITLE: System and method for obtaining scratch registers in computer executable binaries

PUBLICATION-DATE: June 27, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Gillies, David M.	Bellevue	WA	US	
Chaiken, Ronnie	Woodinville	WA	US	
Liu, Jiyang	Sammamish	WA	US	

US-CL-CURRENT: 717/168

ABSTRACT:

A system and method for obtaining scratch registers in a computer-executable binary is provided. Register allocation requests in a computer-executable binary are discovered. In one method, the register allocations are examined procedure-by-procedure. The maximum number of registers requested by any instruction in the procedure is discovered. Then, register requests in the procedure are modified to request the maximum number discovered plus a number of scratch registers. In another method, the register allocations are examined block-by block within a procedure. Dominating register allocations for each block are found. Then the dominating register allocations are modified to request scratch registers.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | In

---

☐ 5. Document ID: US 6434620 B1

L12: Entry 5 of 6   File: USPT   Aug 13, 2002

US-PAT-NO: 6434620
DOCUMENT-IDENTIFIER: US 6434620 B1

TITLE: TCP/IP offload network interface device

DATE-ISSUED: August 13, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Boucher; Laurence B.	Saratoga	CA		
Blightman; Stephen E. J.	San Jose	CA		
Craft; Peter K.	San Francisco	CA		
Higgen; David A.	Saratoga	CA		
Philbrick; Clive M.	San Jose	CA		
Starr; Daryl D.	Milpitas	CA		

US-CL-CURRENT: 709/230; 709/250

ABSTRACT:

An intelligent network interface card (INIC) or communication processing device (CPD) works with a host computer for data communication. The device provides a fast-path that avoids protocol processing for most messages, greatly accelerating data transfer and offloading time-intensive processing tasks from the host CPU. The host retains a fallback processing capability for messages that do not fit fast-path criteria, with the device providing assistance such as validation even for slow-path messages, and messages being selected for either fast-path or slow-path processing. A context for a connection is defined that allows the device to move data, free of headers, directly to or from a destination or source in the host. The context can be passed back to the host for message processing by the host. The device contains specialized hardware circuits that are much faster at their specific tasks than a general purpose CPU. A preferred embodiment includes a trio of pipelined processors devoted to transmit, receive and utility processing, providing full duplex communication for four Fast Ethernet nodes.

6 Claims, 59 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 40

| Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw. Desc | Ir

☐ 6. Document ID: US 6427234 B1

L12: Entry 6 of 6   File: USPT   Jul 30, 2002

US-PAT-NO: 6427234
DOCUMENT-IDENTIFIER: US 6427234 B1
**** See image for Certificate of Correction ****

TITLE: System and method for performing selective dynamic compilation using run-time information

DATE-ISSUED: July 30, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chambers; Craig	Seattle	WA		
Eggers; Susan J.	Seattle	WA		
Grant; Brian K.	Shoreline	WA		
Mock; Markus	Seattle	WA		
Philipose; Matthai	Seattle	WA		

US-CL-CURRENT: 717/140; 717/148, 717/153

ABSTRACT:

Selective dynamic compilation of source code is performed using run-time information. A system is disclosed that implements a declarative, annotation based dynamic compilation of the source code, employing a partial evaluation, binding-time analysis (BTA), and including program-point-specific polyvariant division and specialization and dynamic versions of traditional global and peephole optimizations. The system allows programmers to declaratively specify policies that govern the aggressiveness of specialization and caching, providing fine control over the dynamic compilation process. The policies include directions for controlling specialization at promotion points and merge points, and further define caching policies, and speculative-specialization policies. The system also enables programmers to specialize programs across arbitrary edges, both at traditional locations, such as procedure boundaries, but also within procedures. Programmers are enabled to conditionally specialize programs based on evaluation of arbitrary compile-time and run-time conditions.

20 Claims, 32 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 22

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw. Desc | Ir

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Terms	Documents
L11 and index$	6

Display Format: - Change Format

Previous Page Next Page Go to Doc#

PORTAL

US Patent & Trademark Office

Search: ● The ACM Digital Library ○ The Guide

+scratch +register +patch +allocation

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **scratch** **register** **patch** **allocation**

Found **61** of **131,734**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

☐ Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 61

Result page: **1** 2 3 4 next

Relevance scale

1 Distributed operating systems

Andrew S. Tanenbaum, Robbert Van Renesse

December 1985 **ACM Computing Surveys (CSUR)**, Volume 17 Issue 4

Full text available: pdf(5.49 MB)

Additional Information: full citation, abstract, references, citings, index terms, review

Distributed operating systems have many aspects in common with centralized ones, but they also differ in certain ways. This paper is intended as an introduction to distributed operating systems, and especially to current university research about them. After a discussion of what constitutes a distributed operating system and how it is distinguished from a computer network, various key design issues are discussed. Then several examples of current research projects are examined in some detail ...

2 Fast, effective code generation in a just-in-time Java compiler

Ali-Reza Adl-Tabatabai, Michał Cierniak, Guei-Yuan Lueh, Vishesh M. Parikh, James M. Stichnoth

May 1998 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1998 conference on Programming language design and implementation**, Volume 33 Issue 5

Full text available: pdf(1.44 MB)

Additional Information: full citation, abstract, references, citings, index terms

A "Just-In-Time" (JIT) Java compiler produces native code from Java byte code instructions during program execution. As such, compilation speed is more important in a Java JIT compiler than in a traditional compiler, requiring optimization algorithms to be lightweight and effective. We present the structure of a Java JIT compiler for the Intel Architecture, describe the lightweight implementation of JIT compiler optimizations (e.g., common subexpression elimination, register allocation, and elim ...

3 Dynamically allocating processor resources between nearby and distant ILP

Rajeev Balasubramonian, Sandhya Dwarkadas, David H. Albonesi

May 2001 **ACM SIGARCH Computer Architecture News , Proceedings of the 28th annual international symposium on Computer architecture**, Volume 29 Issue 2

Full text available: pdf(998.02 KB) Publisher Site

Additional Information: full citation, abstract, references, citings, index terms

Modern superscalar processors use wide instruction issue widths and out-of-order execution in order to increase instruction-level parallelism (ILP). Because instructions must be committed in order so as to guarantee precise exceptions, increasing ILP implies increasing *the sizes of structures such as the register file, issue queue, and reorder buffer. Simultaneously, cycle time constraints limit the sizes of these structures, resulting in conflicting design requirements.*

In ...

4 An extensible editor for a small machine with disk storage
Arthur J. Benjamin
August 1972 **Communications of the ACM**, Volume 15 Issue 8

Full text available: pdf(552.64 KB) Additional Information: full citation, abstract, references

A design philosophy for developing a sophisticated utility program is illustrated by the actual design and implementation of a text editor. A versatile data structure is employed so that only a small number of programmed subroutines are necessary for all types of data manipulation. Such a data structure is described, and its merits are illustrated by the ease with which powerful extensions can be implemented in terms of a few basic editing functions.

Keywords: command processing, context searching, executive program, garbage collection, interpreter, list processing, macro language, paging, parameter substitution, recursion, state table, storage allocation, string manipulation, text editing, virtual memory

5 Building a robust software-based router using network processors
Tammo Spalink, Scott Karlin, Larry Peterson, Yitzchak Gottlieb
October 2001 **ACM SIGOPS Operating Systems Review , Proceedings of the eighteenth ACM symposium on Operating systems principles**, Volume 35 Issue 5

Full text available: pdf(1.49 MB) Additional Information: full citation, abstract, references, citings, index terms

Recent efforts to add new services to the Internet have increased interest in software-based routers that are easy to extend and evolve. This paper describes our experiences using emerging network processors---in particular, the Intel IXP1200---to implement a router. We show it is possible to combine an IXP1200 development board and a PC to build an inexpensive router that forwards minimum-sized packets at a rate of 3.47Mpps. This is nearly an order of magnitude faster than existing pure PC-base ...

6 Poor man's watchpoints
Max Copperman, Jeff Thomas
January 1995 **ACM SIGPLAN Notices**, Volume 30 Issue 1

Full text available: pdf(772.87 KB) Additional Information: full citation, abstract, citings, index terms

Bugs that result from corruption of program data can be very difficult to track down without specialized help from a debugger. If the debugger cannot help the user find the point at which data gets corrupted, the user may have a long iterative debugging task. If the debugger is able to stop execution of the program at the point where data gets corrupted, as with watchpoints (also known as data breakpoints), it may be a very simple task to find a data corruption bug. In this paper, we discuss a m ...

7 Techniques for obtaining high performance in Java programs
Iffat H. Kazi, Howard H. Chen, Berdenia Stanley, David J. Lilja
September 2000 **ACM Computing Surveys (CSUR)**, Volume 32 Issue 3

Full text available: pdf(816.13 KB) Additional Information: full citation, abstract, references, citings, index terms

This survey describes research directions in techniques to improve the performance of programs written in the Java programming language. The standard technique for Java execution is interpretation, which provides for extensive portability of programs. A Java interpreter dynamically executes Java bytecodes, which comprise the instruction set of the Java Virtual Machine (JVM). Execution time performance of Java programs can be improved through compilation, possibly at the expense of portabili ...

Keywords: Java, Java virtual machine, bytecode-to-source translators, direct compilers, dynamic compilation, interpreters, just-in-time compilers

8 Sharing and protection in a single-address-space operating system
Jeffrey S. Chase, Henry M. Levy, Michael J. Feeley, Edward D. Lazowska
November 1994 **ACM Transactions on Computer Systems (TOCS)**, Volume 12 Issue 4
Full text available: pdf(2.87 MB) Additional Information: full citation, abstract, references, citings, index terms

This article explores memory sharing and protection support in Opal, a single-address-space operating system designed for wide-address (64-bit) architectures. Opal threads execute within protection domains in a single shared virtual address space. Sharing is simplified, because addresses are context independent. There is no loss of protection, because addressability and access are independent; the right to access a segment is determined by the protection domain in which a thread executes. T ...

Keywords: 64-bit architectures, capability-based systems, microkernel operating systems, object-oriented database systems, persistent storage, protection, single-address-space operating systems, wide-address architectures

9 VCODE: a retargetable, extensible, very fast dynamic code generation system
Dawson R. Engler
May 1996 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1996 conference on Programming language design and implementation**, Volume 31 Issue 5
Full text available: pdf(1.34 MB) Additional Information: full citation, abstract, references, citings, index terms

Dynamic code generation is the creation of executable code at runtime. Such "on-the-fly" code generation is a powerful technique, enabling applications to use runtime information to improve performance by up to an order of magnitude [4, 8,20, 22, 23].Unfortunately, previous general-purpose dynamic code generation systems have been either inefficient or non-portable. We present VCODE, a retargetable, extensible, very fast dynamic code generation system. An important feature of VCODE is that it ge ...

10 Exokemel: an operating system architecture for application-level resource management
D. R. Engler, M. F. Kaashoek, J. O'Toole
December 1995 **ACM SIGOPS Operating Systems Review , Proceedings of the fifteenth ACM symposium on Operating systems principles**, Volume 29 Issue 5
Full text available: pdf(2.16 MB) Additional Information: full citation, references, citings, index terms

11 A comparative performance evaluation of write barrier implementation
Antony L. Hosking, J. Eliot B. Moss, Darko Stefanovic
October 1992 **ACM SIGPLAN Notices , conference proceedings on Object-oriented programming systems, languages, and applications**, Volume 27 Issue 10
Full text available: pdf(2.48 MB) Additional Information: full citation, references, citings, index terms

12 Fast detection of communication patterns in distributed executions
Thomas Kunz, Michiel F. H. Seuren
November 1997 **Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research**
Full text available: pdf(4.21 MB) Additional Information: full citation, abstract, references, index terms

Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time diagrams are often used to obtain a better understanding of the execution of the application. The visualization tool we use is Poet, an event tracer developed at the University of Waterloo. However, these diagrams are often very complex and do not provide the user with the desired overview of the application. In our experience, such tools display repeated occurrences of non-trivial commun ...

13 Associative and Parallel Processors
Kenneth J. Thurber, Leon D. Wald
December 1975 **ACM Computing Surveys (CSUR)**, Volume 7 Issue 4
Full text available: pdf(2.62 MB) Additional Information: full citation, references, citings, index terms

14 Kernel Korner: Dynamic Kernels - Modularized Device Drivers
March 1996 **Linux Journal**
Full text available: html(29.61 KB) Additional Information: full citation, index terms

15 Description-driven code generation using attribute grammars
Mahadevan Ganapathi, Charles N. Fischer
January 1982 **Proceedings of the 9th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**
Full text available: pdf(988.59 KB) Additional Information: full citation, abstract, references, citings

The instruction-set of a target architecture is represented as a set of attribute-grammar productions. A code generator is obtained automatically for any compiler using attributed parsing techniques. A compiler built on this model can automatically perform most popular machine-dependent optimizations, including peephole optimizations. The code generator is also easily retargetable to different machine architectures.

16 The Flux OSKit: a substrate for kernel and language research
Bryan Ford, Godmar Back, Greg Benson, Jay Lepreau, Albert Lin, Olin Shivers
October 1997 **ACM SIGOPS Operating Systems Review , Proceedings of the sixteenth ACM symposium on Operating systems principles**, Volume 31 Issue 5
Full text available: pdf(2.47 MB) Additional Information: full citation, references, citings, index terms

17 DISE: a programmable macro engine for customizing applications
Marc L. Corliss, E. Christopher Lewis, Amir Roth
May 2003 **ACM SIGARCH Computer Architecture News , Proceedings of the 30th annual international symposium on Computer architecture**, Volume 31 Issue 2
Full text available: pdf(335.30 KB) Additional Information: full citation, abstract, references

Dynamic Instruction Stream Editing (DISE) is a cooperative software-hardware scheme for efficiently adding customization functionality---e.g, safety/security checking, profiling, dynamic code decompression, and dynamic optimization---to an application. In DISE, application customization functions (ACFs) are formulated as rules for macro-expanding certain instructions into parameterized instruction sequences. The processor executes the rules on the fetched instructions, feeding the executi ...

18 Dynamo: a transparent dynamic optimization system
Vasanth Bala, Evelyn Duesterwald, Sanjeev Banerjia
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation**, Volume 35 Issue 5
Full text available: pdf(156.03 KB) Additional Information: full citation, abstract, references, citings, index terms

We describe the design and implementation of Dynamo, a software dynamic optimization system that is capable of transparently improving the performance of a native instruction stream as it executes on the processor. The input native instruction stream to Dynamo can be dynamically generated (by a JIT for example), or it can come from the execution of a statically compiled native binary. This paper evaluates the Dynamo system in the latter, more challenging situation, in order to emphasize the ...

19 An overview of the ISPL computer systems design
R. M. Balzer
February 1973 **Communications of the ACM**, Volume 16 Issue 2
Full text available: pdf(668.42 KB) Additional Information: full citation, abstract, references, citings

This paper explores the advantages of the concurrent design of the language, operating system, and machine (via microcode) to create an interactive programming laboratory. It describes the synergistic effect that the freedom to move and alter features from one of these domains to another has had on the design of this system (which has not been implemented). This freedom simplified both incremental compilation and the system's addressing structure, and centralized the communication mechanism ...

Keywords: concurrent design, debugging, hierarchical subsystems, incremental compilation, interprogram communication, operating-system, scheduling, virtual addressing

20 Efficient software-based fault isolation
Robert Wahbe, Steven Lucco, Thomas E. Anderson, Susan L. Graham
December 1993 **ACM SIGOPS Operating Systems Review , Proceedings of the fourteenth ACM symposium on Operating systems principles**, Volume 27 Issue 5
Full text available: pdf(1.49 MB) Additional Information: full citation, abstract, references, citings, index terms

One way to provide fault isolation among cooperating software modules is to place each in its own address space. However, for tightly-coupled modules, this solution incurs prohibitive context switch overhead. In this paper, we present a software approach to implementing fault isolation within a single address space.Our approach has two parts. First, we load the code and data for a distrusted module into its own *fault do main*, a logically separate portion of the application's address space ...

Results 1 - 20 of 61 Result page: **1** 2 3 4 next

CiteSeer Find: scratch register Documents Citations

Searching for **PHRASE scratch register**.
Restrict to: Header Title Order by: Expected citations Hubs Usage Date Try: Amazon B&N Google (RI) Google (Web) CSB DBLP
7 documents found. **Order: number of citations.**

Fast Context Switches: Compiler and Architectural Support .. - Snyder, Whalley, Baker (Correct) (1 citation)
on a live range that has been allocated to a **scratch register**. 2 The **scratch register** can be remapped to a been allocated to a **scratch register**. 2 The **scratch register** can be remapped to a nonscratch register if
www.cs.fsu.edu/~whalley/papers/mnm95.ps

PhysComp96 - Full Paper Draft (Correct)
F -is done to restore the internally used "**scratch register** "C and input "register" X. Input X is
home.attbi.com/~priyadarsan.patra/./conser-b.pdf

A Framework for Conservative and Delay-insensitive Computing - Patra, Fussell (Correct)
is done to restore the internally used "**scratch register**" C and input "register" X. ffl Input X is
ftp.cs.utexas.edu/pub/techreports/tr95-10.ps.Z

Poor Man's Watchpoints - Copperman, Thomas (1994) (Correct)
register, fp, for use by tools that need a **scratch register** (such as the profiler)When the user sets a
ftp.cse.ucsc.edu/pub/tr/ucsc-crl-94-17.ps.Z

Writing Efficient Programs for the Motorola M.CORE Architecture - Taimur Asl Am (Correct)
(EPC0C48,0hF9a-T0519SScratch (SS0-SS4)be"d byto }e"
www.esconline.com/db_area/98spring/226.pdf

Digital Equipment Corporation Hudson, Massachusetts - Us Et Ts (Correct)
register, v0 is the return value, t0 is a **scratch register** (caller saved)and s0s2 are preserved registers can be more expensive than **scratch registers** because they must be saved and re stored
ftp.digital.fr/pub/DEC/Micro29.ps

Translating Verdi Intermediate Language Into SPARC Code - Meisels (1994) (Correct)
zero value and data sink 1 g1,scr1 **scratch register** 2 g2 temporary 3 g3 temporary 4 g4 pointer 15 o7,scr2 return address of callee/**scratch register** 16 l0 local or temporary 17 l1 local or
ftp.ora.on.ca/pub/doc/94-5463-12.ps.Z

Try your query at: Amazon Barnes & Noble Google (RI) Google (Web) CSB DBLP

YAHOO! search scratch register Yahoo! Search Advanced Preferenc

Web Images Directory Yellow Pages News Products

powered by hp

TOP 20 WEB RESULTS out of about 1,170,000. Search took 0.14 seconds. (What's this?)

1. Look RS232 - RS 232 (serial port) programming
   Look RS232 can help you to debug your program connected to an RS232 external device. ... LCR. LSR. MCR. **Scratch Register**. Interfacing devices. Waveforms. Level converters ... **Scratch Register**. The **scratch register** is not used for connection, it is rather used as a place to leave a ...
   www.lookrs232.com/rs232/scratch_register.htm - 16k - Cached

2. http://www.cs.caltech.edu/~miandai/cs134a/linux/include/asm-ia64/sn/sn1/hublb.h
   ... define LB_**SCRATCH**_REG0 0x00608000 /* **Scratch Register** 0 */ #define LB_**SCRATCH**_REG1 0x00608008 /* **Scratch Register** 1 */ #define LB_**SCRATCH** ...
   www.cs.caltech.edu/~miandai/cs134a/linux/include/asm-ia64/sn/sn1/hublb.h - 60k - Cached - More pages from this site

3. OPT_GenericStackManager
   ... Class to represent a physical **register** currently allocated as a **scratch register**. ... be used as a **scratch register** to hold **register** r Except, do NOT return any **register** that ...
   oss.software.ibm.com/developerworks/oss/jikesrvm/api/com/ibm/JikesRVM/opt/OPT_GenericStackManager.ht ... - 112k - Cached

4. **Register** Allocation
   ... the spilled value into a **scratch register**. On PowerPC, the original **register** allocator reserved three ... use as **scratch**, so finding a free **scratch register** was almost always trivial ...
   www.usenix.org/events/jvm02/full_papers/alpern/alpern_html/node15.html - 11k - Cached

5. http://search.cpan.org/src/JGOFF/parrot-0.0.8.1/jit/arm/jit_emit.h
   ... endif /* Registers * * r0 Argument/result/**scratch register** 0. * r1 Argument/result/**scratch register** 1. * r2 Argument/result/**scratch register** 2 ...
   search.cpan.org/src/JGOFF/parrot-0.0.8.1/jit/arm/jit_emit.h - 37k - Cached

6. http://cs.nyu.edu/~yap/unsup/unsup/installers/exact/gcc/gcc-3.1/gcc/config/m68hc11/m68hc11.md
   ... for 68hc12 ;; (used for **scratch register**) ;; w **register** 'sp' 16-bit ;; x **register** 'x' 16-bit ;; y ... Represents the temporary/**scratch register** *_.tmp ;; The **scratch register** is used in ...
   cs.nyu.edu/~yap/unsup/unsup/installers/exact/gcc/gcc-3.1/gcc/config/m68hc11/m68hc11.md - 194k - Cached

7. http://www.pa.msu.edu/hep/d0/ftp/l1/framework/l3_interface/dave_init_test_1.rio
   ! Initial attempt at setting up the DAVE L3 <-> TFW Interface ! Created 11-APR-02 D.E. in the **scratch** directory ! Setup the Board Support Functions FPGA (BSF) on each card. !
   www.pa.msu.edu/hep/d0/ftp/l1/framework/l3_interface/dave_init_test_1.rio - 12k -

Cached

8. http://home.maine.rr.com/russell316/3DO/txt/apcs.txt
... 1 / integer result / **scratch register** a2 1 argument 2 / **scratch register** a3 2 ... 3 / **scratch register** a4 3 argument 4 / **scratch register** v1 4 **register** variable v2 5 **register** variable ...
home.maine.rr.com/russell316/3DO/txt/apcs.txt - 47k - Cached

9. Porting SBR-Threading
... The ARG-**register** should be a **scratch-register** in the local ABI (application binary interface - the ... cpu ABI do know some **scratch register** for local computations whose value ...
pfe.sourceforge.net/manual/ch01s11.html - 14k - Cached - More pages from this site

10. http://www.cs.cmu.edu/~jh6p/classes/studio/gcc-2.5.8/config/a29k/a29k.md
... is a **scratch** general **register** and operand 3 is a **scratch register** ;; used for BP ... is a **scratch** general **register** and operand 3 is a **scratch register** ;; used for BP ...
www.cs.cmu.edu/~jh6p/classes/studio/gcc-2.5.8/config/a29k/a29k.md - 84k - Cached - More pages from this site

11. http://www.egr.msu.edu/classes/ece482/Teams/98fall/design2/deliverables/resources/LCD/gcc-2.8.1/conf ig/a29k/a29k.md
... is a **scratch** general **register** and operand 3 is a **scratch register** ;; used for BP ... is a **scratch** general **register** and operand 3 is a **scratch register** ;; used for BP ...
www.egr.msu.edu/classes/ece482/Teams/98fall/design2/deliverables/resources/LCD/gcc-2.8.1/config/a29k ... - 88k - Cached

12. 2.4.19-46um/include/asm-ia64/ptrace.h
... bits for **scratch** registers such that bit N==1 iff **scratch register** rN is a NaT */235 extern ... bits for **scratch** registers such that **scratch register** rN is a NaT ...
www.cs.utexas.edu/users/ygz/378-03S/lxr/source/include/asm-ia64/ptrace.h - More pages from this site

13. http://www.circlemud.org/pub/dev3210/gcc/dsp3210.c
/* Assembly language support routines for the DSP3210.
www.circlemud.org/pub/dev3210/gcc/dsp3210.c

14. http://www.ece.utexas.edu/projects/ee380l/downloads/gcc-2.95.1/gcc/config/dsp16xx/dsp16xx.md
;;- Machine description for the AT&T DSP1600 for GNU C compiler ;; Copyright (C) 1994, 1995, 1997, 1998 Free Software Foundation, Inc. ;; Contributed by Michael Collison (collison@world.std.com). ;; This file is part of GNU CC. ;;
www.ece.utexas.edu/projects/ee380l/downloads/gcc-2.95.1/gcc/config/dsp16xx/dsp16xx.md - 57k - Cached - More pages from this site

15. http://www.thorkildsen.no/faqsys/docs/oak.txt
... pixelport) 3DEh index 09h (R/W): **Scratch Register** 1 bit 0 Set if 24bit ... index 0Ah (R/W): **Scratch Register** 2 3DEh index 0Bh (R/W): **Scratch Register** 3 bit ...
www.thorkildsen.no/faqsys/docs/oak.txt - 23k - Cached - More pages from this site

16. Body
... Argument 1/integer result/**scratch register**. Caller-saved registers - subprogram can use them as **scratch** registers, but it ... R14. lr. link **register/scratch register**.

Receives return addr ...
www.cs.uni.edu/~fienup/cs041s03/lectures/lec20_3-11-03.htm - 9k - Cached

17. Programmed I/O (PIO)
Programmed I/O (PIO) The udi_cmos driver uses PIO to read and write values on the CMOS RAM device. This is done by first specifying the I/O characteristics of the device. " cmos_udi.c PIO device properties" /* *
docsrv.sco.com/UDI_dwg/dwg_pio.html - 15k - Cached

18. http://www.klid.dk/ftp/gnu/gnu-utils/gnu/gcc-2721/local-alloc.c
/* Allocate registers within a basic block, for GNU compiler.
www.klid.dk/ftp/gnu/gnu-utils/gnu/gcc-2721/local-alloc.c

19. http://www-wjp.cs.uni-sb.de/projects/verification/PVS/software/dlx_rte/dlxgcc/config/mips/mips.md
;; Mips.md Machine Description for MIPS based processors ;; Contributed by A. ... instruction(s) ;; move data movement within same **register** set ;; xfer transfer to/from coprocessor ;; hilo ... We mark the **scratch register** as early clobbered to prevent this. (define ...
www-wjp.cs.uni-sb.de/projects/verification/PVS/software/dlx_rte/dlxgcc/config/mips/mips.md - 201k - Cached - More pages from this site

20. http://wwwbzs.tu-graz.ac.at/~fleischh/computer/hp48/horn/programr/prog_pc/hp.prg/sasm.txt
... 25 6.11.5 **Scratch Register** Instructions...... 25 6.11.6 Data Pointer Instructions ... C), (C,A), (D,C) }. ss Represents a **scratch register** name (R0, R1, R2, R3, or R4). dp ...
wwwbzs.tu-graz.ac.at/~fleischh/computer/hp48/horn/programr/prog_pc/hp.prg/sasm.txt - 294k - Cached - More pages from this site

Results Page:
1 2 3 4 5 6 7 8 9 10 ▶ **Next**

Help us improve your search experience. Send us feedback.

Web | Images | Directory | Yellow Pages | News | Products

Your Search: scratch register | Yahoo! Search | Advanced Web Search Preferences

Yahoo! Search is hiring! Learn about job opportunities
Save time with the Yahoo! Search Toolbar

Y! | Search Web | Search This Site | Highlight | Bookmarks

powered by hp